Amendments to the Claims

This listing of claims will replace all prior versions and listings of claims in the application.

1. (currently amended) A method for decreasing a computer application start-up time, ~~the application being compiled to create a first object code file loaded into a runtime environment and creating a first application state,~~ comprising:

compiling application source code into a first object file for a runtime environment, the first object code file including application objects, each marked with a unique identifier, and instructions defining relations between the application objects and built-in objects in the runtime environment;

processing the instructions to create relations between the application objects and built-in objects;

generating initialization code representing a first application state;

creating a serialized representation of ~~application objects in~~ the initialization code ~~runtime environment~~;

building an optimized object code file using the serialized representation and the first object code file, wherein the step of building includes: ~~wherein the optimized object code file includes instructions creating relations between objects in the runtime environment; and~~

identifying each application object in the serialized representation that has a unique identifier referring to an application object in the first object code file; and

for each identified application object, copying the application object in the initialization code with the same unique identifier to the optimized object code file;

loading the optimized object code file into a new runtime environment to create a second application state isomorphic to the first application state.; and

executing the optimized object code in the new runtime environment.

2. (currently amended) The method of claim 1 wherein the step of creating a serialized representation includes:

reading from a runtime memory space a description of each application object of a running computer application.

3. (previously presented) The method of claim 1 wherein the step of creating a serialized representation includes enumerating a description of each object of the computer application using reflection.

4. (original) The method of claim 2 wherein the runtime environment is a virtual machine.

5. (previously presented) The method of claim 4 wherein the virtual machine is a presentation renderer.

6. (currently amended) The method of claim 1 wherein the step of creating a serialized representation comprises:

identifying each application object of a running computer application by a unique identifier.

7. (previously presented) The method of claim 6 wherein the step of creating a serialized representation comprises detaching each object from an object hierarchy and creating a description of each slot in said object.

8. (previously presented) The method of claim 6 wherein the step of creating a serialized representation includes the step of providing the serialized representation directly to the building step.

9. (previously presented) The method of claim 1 wherein the serialized representation of application objects in a runtime environment is written in an Extensible Markup Language data format.

10. (previously presented) The method of claim 9 wherein the step of creating a serialized representation comprises storing a markup language file prior to said step of building.

11. (previously presented) The method of claim 1 wherein the step of creating a serialized representation comprises assigning a serialization identifier to each object.

12. (previously presented) The method of claim 1 further including the step of developing the computer application in an interpreted language.

13. (previously presented) The method of claim 6 wherein the step of creating a serialized representation comprises:

assigning a function ID to each function in the computer application.

14. (original) The method of claim 13 wherein the method further includes:

creating a function ID table associating each function ID with function code.

15. (original) The method of claim 13 wherein the method further includes assigning unique function identifiers to functions within closures.

16. (cancelled)

17. (previously presented) The method of claim 1 wherein the step of creating a serialized representation comprises:

writing the serialized representation to a text file prior to said step of building.

18. (cancelled)

19. (previously presented) The method of claim 1 wherein the step of creating a serialized representation is performed in a different runtime.

20. (currently amended) A method for providing an optimized application, comprising:

compiling an application provided in a source language to create a first object code file containing application objects, each application object being marked with a unique identifier;

initializing a first memory space that represents a first program state that contains built-in objects of a runtime environment;

processing instructions in the application objects in the first object code file in order to add objects and create relations within the first memory space to create a second memory state that represents a second application state;

executing portions of the application marked for execution to create a third memory state that represents a third application state;

~~initializing the first object code file in a runtime environment to create a first application state; and~~

creating a serialized representation of the third application state ~~application objects in the runtime environment~~; and

building an optimized object code file using the serialized representation and the first object code file, ~~wherein the optimized object code file includes instructions creating relation relations between objects in the runtime environment to create a second application state isomorphic to the first application state~~, the step of building includes:~~.~~

identifying each application object in the serialized representation that has a unique identifier referring to an application object in the first object code file; and

for each identified application object, copying the application object in the third application state with the same unique identifier to the optimized object code file.

21. (previously presented) The method of claim 20 wherein the step of creating comprises:

reading from the runtime environment a description of each object of the application.

22. (original) The method of claim 21 wherein the step of creating further comprises:

outputting the description to a rebuilder.

23. (previously presented) The method of claim 22 wherein the step of outputting comprises storing the serialized representation in a text file and providing the text file to the rebuilder.

24. (previously presented) The method of claim 22 wherein the step of outputting comprises:

writing the description to an Extensible Markup Language file and providing the Extensible Markup Language file to the rebuilder.

25. (original) The method of claim 20 wherein the runtime environment is a virtual machine.

26. (previously presented) The method of claim 25 wherein the virtual machine is a presentation renderer.

27. (original) The method of claim 20 wherein the step of creating includes assigning a serialization identifier to each initialized object.

28. (original) The method of claim 20 wherein the step of creating includes the steps of enumerating each object in a global scope and writing a serialized description of each said object.

29. (previously presented) The method of claim 20 wherein the step of creating comprises:

assigning a function ID to each function in the application.

30. (original) The method of claim 29 wherein the method further includes:

creating a function ID table associating each function ID with a function call.

31. (original) The method of claim 29 wherein the method further includes assigning function identifiers to functions within closures.

32. (canceled)

33. (currently amended) A method of operating an application, comprising:

requesting an application from an application source server;

receiving a first object code file loaded into a runtime environment and creating a first application state;

executing portions of the application marked for execution and creating a second application state;

receiving an optimized object code file using a serialized description of the second application state ~~from the application source server~~ and the first object code file, the optimized object code file including instructions creating relations between objects in the runtime environment, the optimized object code built by; ~~and~~

identifying each application object in the serialized representation that has a unique identifier referring to an application object in the first object code file;

for each identified application object, copying the application object in the second application state with the same unique identifier to the optimized object code file; and

loading the optimized object code file into a new runtime environment to create a second application state isomorphic to the first application state.

34. (previously presented) The method of claim 33 wherein the first object code file includes media assets.

35. (currently amended) The method of claim 33 wherein the runtime environment ~~memory state~~ is ~~of~~ a presentation renderer.

36. (cancelled)

37. (cancelled)

38. (currently amended) One or more processor readable storage devices having processor readable code embodied on said one or more processor readable storage devices,

said processor readable code for programming one or more processors to perform a method for decreasing a computer application start-up time, the computer application being compiled to create a first object code file loaded into a runtime environment and creating a first application state, comprising the steps of:

executing portions of the application marked for execution to create a second application state;

creating a serialized representation of the second application state ~~objects in the runtime environment~~;

building an optimized object code file using the serialized representation of the application objects and the first object code file, wherein the optimized object code file includes application objects, each marked with a unique identifier, so that corresponding application objects in the first application state can be identified, and copying application objects from the second application state to the optimized object code file based on the unique identifiers ~~instructions creating relations between objects in the runtime environment~~; and

loading the optimized object code file into a new runtime environment to create a ~~second~~ third application state isomorphic to the ~~first~~ second application state.

39. (original) One or more processor readable storage devices as described in claim 38 wherein the step of creating includes:

reading from a runtime environment memory space a description of each object of a running application.

40. (previously presented) One or more processor readable storage devices as described in claim 38 wherein the step of creating includes enumerating a description of each object of the computer application using reflection.

41. (original) One or more processor readable storage devices as described in claim 38 wherein the step of creating comprises:

identifying each object of a running application by a unique identifier.

42. (previously presented) One or more processor readable storage devices as described in claim 41 wherein the step of creating comprises:

writing a description to a text file and compiling the text file.

43. (previously presented) One or more processor readable storage devices as described in claim 41 wherein the step of creating comprises:

writing a description to an Extensible Markup Language file and compiling a markup language file.

44. (original) One or more processor readable storage devices as described in claim 41 wherein the step of creating comprises detaching each object from an object hierarchy and creating a description of each slot in said object.

45. (previously presented) One or more processor readable storage devices as described in claim 41 wherein the step of creating further includes the steps of:

determining whether the object is a class; and

writing a serialized description of the class.

46. (previously presented) One or more processor readable storage devices as described in claim 39 wherein the serialized representation of application objects in a runtime environment is written in an Extensible Markup Language data format.

47. (original) One or more processor readable storage devices as described in claim 39 wherein the step of creating comprises assigning a serialization identifier to each object.

48. (previously presented) One or more processor readable storage devices as described in claim 39 further including the step of developing the computer application in an interpreted language.

49. (previously presented) One or more processor readable storage devices as described in claim 41 wherein the step of creating comprises:

assigning a function ID to each function in the computer application.

50. (original) One or more processor readable storage devices as described in claim 49 wherein the method further includes:

creating a function ID table associating each function ID with function code.

51. (original) One or more processor readable storage devices as described in claim 49 wherein the method further includes assigning function identifiers to functions within closures.

52. (currently amended) A method for reducing the start-up time of an application, comprising:

compiling the application into a first object code file;

loading the first object code file into a first runtime environment to create a first application state;

executing portions of the application marked for execution to create a second application state;

creating a serialized representation of the second application state ~~a memory space in said first runtime environment~~;

building a second object code file using said serialized representation and the first object code file, wherein application objects from the second application state are copied to the second object code file based on a unique identifier associated with each application object in the serialized representation ~~includes instructions creating relations between objects in the runtime environment~~; and

loading said second object code into a second runtime environment to create a third ~~second~~ application state isomorphic to the first application state.

53. (original) The method of claim 52 wherein the step of compiling is performed on an interpreted language application.

54. (original) The method of claim 52 wherein the step of creating is performed by calling at least one function from said first runtime environment.

55. (original) The method of claim 52 wherein the step of creating is performed in the same runtime environment as said application.

56. (original) The method of claim 52 wherein the step of creating is performed in a different runtime environment from said application.

57. (cancelled)

58. (cancelled)

59. (canceled)

60. (cancelled)

61. (cancelled)

62. (currently amended) A method for delivering an application via a network, the application being compiled to create a first object code file loaded into a runtime environment and creating a first application state, the method comprising:

creating relations between application objects in the in first object code file an built-in objects in the runtime environment to create a second application state;

executing portions of the application marked for execution to create a third application state;

creating a serialized representation of the third application state ~~application objects in the runtime environment~~;

building an optimized object code file using the serialized representation and the first object code file, wherein application objects are copied from the third application state into the optimized object code file based on a unique identifier associated with each application object in the serialized representation ~~includes instructions creating relations between objects in the runtime environment~~; and

loading the optimized object code file into a new runtime environment via the network to create a ~~second~~ fourth application state isomorphic to the ~~first~~ third application state.

63. (previously presented) The method of claim 62 wherein the step of creating includes enumerating a description of each object of the application using reflection.

64. (original) The method of claim 63 wherein the new runtime environment is a virtual machine.

65. (previously presented) The method of claim 64 wherein the virtual machine is a presentation renderer.